EPJ Web of Conferences **226**, 01002 (2020) https://doi.org/10.1051/epjconf/202022601002
Mathematical Modeling and Computational Physics 2019

Lattice Study of QCD Properties at the "Govorun" Supercomputer

Nikita Astrakhantsev[1,⋆], *Victor* Braguta[1,⋆⋆], *Andrey* Kotov[1,⋆⋆⋆], and *Alexander* Nikolaev[2,⋆⋆⋆⋆]

[1] *Bogoliubov Laboratory of Theoretical Physics, Joint Institute for Nuclear Research, 141980 Dubna, Russia*
[2] *Department of Physics, College of Science, Swansea University, Swansea SA2 8PP, United Kingdom*

Abstract. This report is devoted to Lattice QCD simulations carried out at the "Govorun" supercomputer. The basics of Lattice QCD methodology, the main Lattice QCD algorithms and the most numerically demanding routines are reviewed. We then present details of our multiGPU code implementation and the specifics of its application on the "Govorun" architecture. We show that implementation of very efficient and scalable code is possible and present scalability tests. We finally relate our program with the goals of the NICA project and review the main physical scenarios that are to be studied numerically.

1 Introduction

The Quantum Chromodynamics (QCD) is the part of the Standard model describing such elementary particles as quarks and gluons. The main feature of the QCD is the confinement/deconfinement phase transition [1]. In short, at high energies (high temperature T or high particle density n) the quarks and gluons are "good" degrees of freedom and the quark-gluon matter can be considered as consisting of independent quarks and gluons mildly interacting with each other. However, at low energies the quarks and gluons form hadrons (protons, neutrons, *etc.*) which become the relevant degrees of freedom instead of the independent quarks and gluons. This significant change in the relevant degrees of freedom is of paramount theoretical importance and is being studied by many researchers.

Even though it is believed that QCD is the correct theory to describe the strong sector of the Standard model, the QCD Lagrangian is very complicated and notoriously strongly coupled. The α_s constant describing the interaction strength between quarks and gluons is large ($\alpha_s \sim 1$) in many physically relevant scenarios. This does not allow the expansion in α_s series and the derivation of experimentally observable predictions analytically from the QCD first principles. However, the lattice simulations of QCD enable the study of a lot of QCD properties using modern supercomputers [2].

Currently, several heavy-ion collision experiments such as NICA, RHIC, LHC and FAIR are aimed at the experimental study of the QCD phase diagram in the temperature-baryon density (T, μ) plane[1]. As already noticed, the only reliable way based on the first principles to the study of the QCD properties in the (T, μ) plane is provided by the lattice simulations of QCD (LQCD). The method is based

⋆e-mail: nikita.astronaut@gmail.com
⋆⋆e-mail: braguta@itep.ru
⋆⋆⋆e-mail: kotov.andrey.yu@gmail.com
⋆⋆⋆⋆e-mail: aleksandr.nikolaev@swansea.ac.uk

[1] Typically, the baryon density is parameterized by the baryon chemical potential μ.

on the Feynman path integral technique and is widely known to be reliable. Many experimental observables were confirmed by LQCD direct computations. Due to LQCD we know a lot about the thermodynamic properties of the QCD at zero or small baryon density [3, 4].

Although the LQCD method is reliable in terms of controllable approximations, it is numerically very demanding and requires novel algorithms and hardware. In the recent years it was noticed that GPU (Graphics Processing Unit) can be utilized to significantly speed-up the LQCD simulations. One of the last trends in LQCD is the application of multiGPU techniques. In this paper we will review the multiGPU algorithms that we have implemented. We will discuss the basic speed-up techniques and the multiGPU specifics. We will also discuss the perspectives of the "Govorun" supercomputer in case of multiGPU simulations and perform a scalability analysis of our code. Lastly, we will discuss the main projects that require the multiGPU code and their relation to the NICA collider.

2 Lattice QCD methodology

2.1 Lattice action and discretization

The QCD Lagrangian is well-known and gives accurate analytical predictions where possible. However, in many cases direct QCD calculations from the first principles are only possible in the LQCD. In LQCD [5] one starts with the introduction of discrete coordinates $x_\mu(n_\nu)$, $n_\nu \in 0, 1, 2, \ldots$ called the Lattice discretization. The fields are now defined only at the finite number of lattice sites. Usually the lattice has $L_s \sim 64$ steps in every spatial direction and $L_t \sim 64$ sites in the temporal direction. The introduced lattice has a lattice spacing a and in the continuum limit $a \to 0$ should give the experimentally observable predictions. The discretized LQCD action reads

$$\mathcal{L}_{LQCD} = a^4 \frac{1}{3} \sum_{x\in\Lambda,\ \mu<\nu} \mathrm{ReTr}\left(\hat{1} - P_{\mu\nu}(x)\right) - a^4 \sum_{x\in\Lambda} \bar{q}(x)(\hat{D}(U) + m)q(x),$$

where $P_{\mu\nu}(x) = U_\mu(x)U_\nu(x+\mu)U^\dagger_\mu(x+\nu)U^\dagger_\nu(x)$ is the plaquette, $\hat{D}(U)$ is the Dirac operator and Λ is the whole set of $L_s^3 \times L_t$ lattice sites. In the Lattice version of the QCD Lagrangian, the gluonic fields in $SU(3)$ algebra $A_\mu(x)$ become the $SU(3)$ group $U_\mu(x) = \exp\left(iaA_\mu(x)\right)$, while $q(x)$ are the 3-values complex columns. In the continuum limit the terms above reproduce the well-known expressions:

$$S_G(U) = a^4 \frac{1}{3} \sum_{x\in\Lambda,\ \mu<\nu} \mathrm{ReTr}\left(\hat{1} - P_{\mu\nu}(x)\right)\Big|_{a\to 0} = -\frac{1}{4}\sum_{a=1}^{8} \int d^4x F^{\mu\nu}_a(x) F^a_{\mu\nu}(x),$$

$$a^4 \sum_{x\in\Lambda} \bar{q}(x)(\hat{D}(U) + m)q(x)\Big|_{a\to 0} = \int d^4x\, \bar{q}(x)(\gamma^\mu\partial_\mu + ig\gamma^\mu A_\mu(x) + m)q(x).$$

The Lattice QCD studies the system in thermal equilibrium. The main object of interest is the partition function which contains valuable information about the system properties:

$$\mathcal{Z} = \int \mathcal{D}U\mathcal{D}\bar{q}\mathcal{D}q\ \exp\Big(-S_G(U) + \int d^4x \bar{q}(x)(\hat{D}(U) + m)q(x)\Big).$$

The partition function $\mathcal{Z}$ contains integration over the continuum set of variables which turns out to be the finite set of $L_s^3 \times L_t$ variables within the lattice discretization. The Grassmanian anti-commuting variables $q(x)$ can be integrated out analytically to give

$$\mathcal{Z} = \int \mathcal{D}U\mathcal{D}\bar{q}\mathcal{D}q\ \exp\left(-S_G(U)\right) \times \prod_{f=u,d,s,\ldots} \det\left(\hat{D}_f(U) + m_f\right),$$

where the det operation is taken over an operator of the size $L_s^3 \times L_t \times L_s^3 \times L_t$ and f denotes the present quark flavors. Although the above computation is infeasible on the today computers (we shall discuss that later), we can now review the assumptions made, the approximations and the whole computation pipeline. First, one computes the partition function $\mathcal{Z}(\mu, T, a, m_f, L_s, L_t)$. In order to reproduce the physical result, one should perform the continuum extrapolation $a \to 0$ and the infinite volume extrapolation $L_s, L_t \to \infty$. The uncertainties of the computation can always be reduced to the required level by performing longer simulations. Thus, the method keeps all the uncertainties and approximations under control.

The next step of the methodology is getting rid of the fermionic determinant $\det(\hat{D}_f(U) + m_f)$ to allow numerical study. This is done by application of the well-known Gaussian integration formula

$$\det(\hat{D}(U) + m) = \int \mathcal{D}\bar{\varphi}\mathcal{D}\varphi \exp\left(-\bar{\varphi}\left[\hat{D}(U) + m\right]^{-1}\varphi\right),$$

where $\varphi(x)$ is the $L_s^3 \times L_t$-dimensional complex vector. The $S_F(U,\varphi) = \bar{\varphi}\left[\hat{D}(U) + m\right]^{-1}\varphi$ operation is literally the action of the matrix on the vector and then the scalar product with $\bar{\varphi}$. The partition function now reads

$$\mathcal{Z} \sim \int \mathcal{D}U\mathcal{D}\bar{\varphi}\mathcal{D}\varphi \exp(-S_G(U) - S_F(U,\varphi)).$$

In spite of the fact that we can now simulate this partition function on the lattice, one has to perform numerous inversions of the very large and sometimes badly-conditioned $\left[\hat{D}(U) + m\right]$ operator. This is done by the application of the conjugate gradient method.

2.2 The Hybrid Monte Carlo (HMC) algorithm

The final expression for the partition function in the typical contemporary lattice simulation contains $\sim 10^8$ integration variables. The integral is estimated stochastically within the Monte-Carlo sampling [5], however naive Monte-Carlo sampling fails because the $S(U,\varphi)$ distribution is very localized and field configurations away from the action minimum (maximum of $\exp(-S(U,\varphi))$) are significantly suppressed. In this situation the clever integration technique of [6] is employed. Let us perform the Legendre transform and consider the Hamiltonian

$$H(U,\varphi,\pi) = \int d^4x \left(S_G(U(x)) + S_F(U(x),\varphi(x)) + \frac{1}{2}\pi^2(x)\right).$$

The $\pi(x)$ are the conjugate momenta to the $U(x)$-fields. At every step of the algorithm, one generates the momenta from a normal distribution $\pi(x) \sim N(0,1)$. Then the evolution in the fictive *molecular time* τ is performed in accordance with the Hamilton equations of motion

$$\frac{\partial U(x,\tau)}{\partial \tau} = \frac{\partial H(x,\tau)}{\partial \pi(x,\tau)}; \quad \frac{\partial \pi(x,\tau)}{\partial \tau} = -\frac{\partial H(x,\tau)}{\partial U(x,\tau)}.$$

After several steps of the evolution, the final field configuration U', φ', π' is accepted as the next Monte Carlo sample with the Metropolis probability $P(H \to H') = \min(1, \exp(H' - H))$. At sufficiently large number of iterations the field configurations (U,φ) will be distributed in accordance with the required weight $W(U,\varphi) = \exp(-S(U,\varphi))$. The algorithm can be imagined as a Brownian motion in the highly-dimensional configuration space.

Despite the fact that at this point the algorithm for LQCD is much more feasible, there are still many pending problems. First of all, the Hamilton dynamics itself is described by a very complicated

partial differential equation in 4-dimensional space-time. The variable manipulations involved complicated algebra such as multiplication of complex $SU(3)$ fields and matrix expotentiation $\exp U_\mu(x)$. However, the main numerical effort resides in the computation of the force $\partial H(x,\tau)/\partial U(x,\tau)$ which contains the inverse Dirac operator $\left[\hat{D}(U)+m\right]^{-1}\varphi(x)$.

3 LQCD speed-up algorithms and GPU

3.1 Omelyan integrator and scale separation

The general idea behind the HMC algorithm is to evaluate the field configuration far enough from the previous configuration to consider them independent (non-correlated) Monte Carlo samples. Since the computation of the force at every integration step is very demanding, one needs to reduce the number of force computations for the same length of τ–molecular-time evolution. The novel Omelyan integrator [6] splits the integration time τ into several pieces where the momenta (P) and fields (U) are varied independently

$$T(\tau) = T_P(\lambda\tau)T_U(\tau/2)T_P((1-2\lambda)\tau)T_U(\tau/2)T_P(\lambda\tau).$$

Here $\lambda \approx 0.193$. The method allows to reach an $O(\Delta\tau^3)$ integration error instead of the standard $O(\Delta\tau^2)$. This allows to perform much longer integration trajectories at the cost of smaller force evaluations.

Another strong idea behind speed-up is that the fermionic part of the force $-\partial S_F/\partial U$ varies much slower than the gluonic part $-\partial S_G/\partial U$. So it is possible to keep the fermionic force term constant within several reevaluations of the gluonic force. This allows to reduce the number of $\left[\hat{D}(U)+m\right]^{-1}\varphi(x)$ evaluations within a factor of ~ 3.

3.2 GPU, linear algebra and memory alignment

The LQCD formulation operates essentially with large vectors and operators. The action is written as the sum of terms over lattice sites. The LQCD can be efficiently speed-up by GPU handling. The modern GPUs have $\sim 3\times10^3$ working threads and allow the achievement of drastic speed-up in LQCD simulations. The $\hat{D}(U)+m$ operator action and standard linear algebra (LA) operations such as vector manipulation, scalar products can be efficiently paralellized. It is also worth noting the importance of the contemporary software usage. For instance, the "thrust" package provides efficient scalar product routines for the GPU. The package employs clever reduction algorithm generating minimal overhead. Another example is the "cuBLAS" library which efficiently implements the LA operations.

It is also important to consider the *coalescence issue*. Even though the number of threads on the GPU is $\sim 3\times10^3$, many of them might be stalled because of the memory read latency. In typical LQCD application, each thread reads ~ 400 bytes of data from the GPU global memory. Such reads should be coalesced, so that subsequent threads read subsequent memory addresses (Figure 1). If the reads are coalesced, then the data for several threads is obtained in single read, which reduces the memory bandwidth load. If the reads are not coalesced, only $\sim 10\,\%$ threads were working simultaneously in our computations. However, the coalescence allowed to get rid of the stalled threads.

4 MultiGPU applications and the "Govorun" supercomputer

With the use of supercomputers, larger lattice volumes and smaller lattice spacings a can be explored. However, the HMC algorithm time scales as $O(V^{5/4})$ where V is the lattice volume. Thus, larger

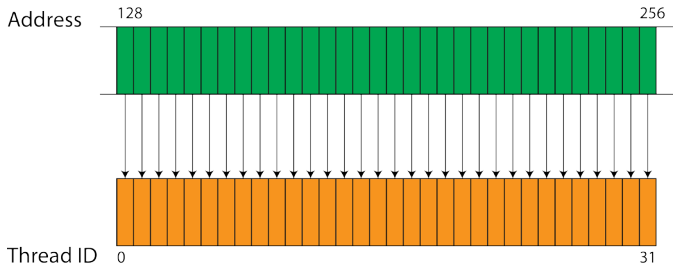

Figure 1. Coalesced memory reading pattern: subsequent threads read subsequent memory addresses.

volumes lead to substantial increase of the simulation times. Moreover, large lattice volumes do not fit into a single-GPU memory. To obviate this circumstances, multiGPU LQCD codes [7] have been developed.

As noted, the inversion of the $\hat{D}(U) + m$ operator is done within the conjugate gradient (CG) method. Each iteration of CG requires 2 scalar products, 3 LA operations and $\hat{D}(U)$ action. The $\hat{D}(U)$ operator is local (Figure 2), so it involves nearest neighbors only in the element computation.

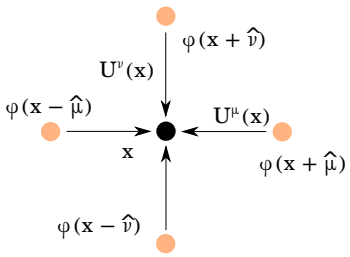

Figure 2. The Dirac operator scheme. Each thread builds the combination of 4×2 gauge fields $U(x)$ and pseudofermion fields $\varphi(x)$.

Based on the Dirac operator structure, the following multiGPU strategy is used (Fig. 3). The lattice is cut along the z–direction (any spatial dimension, because usually $L_s > L_t$) into N equal pieces, where N is the number of available GPUs, so $L_z = L_s/N$. In order to perform the action of $\hat{D}(U)$ at the edge of every GPU memory, we need to know the edge of left and right neighbors, because the $\hat{D}(U)$ connects the site with its nearest neighbors. In order to keep the code simpler, we increase $L_z \to L_z + 2$, so that the $z = 0$ and $z = L_z - 1$ slices would contain the information about the neighbors [8]. These slices are called "buffers" and receive the data from neighbors and store it in order to perform $\hat{D}(U)$ action at the edge. The extension of L_z allows one to keep many functions within the code untouched.

The memory transfer is a costly operation, however in some cases its latency can be completely hidden behind the computation. To perform this transfer-computation overlap, we call the $z = 1$ and $z = L_z - 2$ slices *halos*, while $2 < z < L_z - 2$ slices are the *bulk* (Fig. 3). The black (red color online) color within represents the bulk of the GPUs. These sites can be processed without knowledge about the neighbors. The dark gray (orange online) sites are "halos" – their processing requires data transfer to buffers.

Thus, one performs memory transfer of halos *in parallel* with the computation of Dirac operator on bulk. If the computations are sufficiently long, the memory transfer cost is completely hidden. We also would like to emphasize that contemporary GPUs are designed to perform parallel computations and memory copy, thus the computation speed is not reduced.

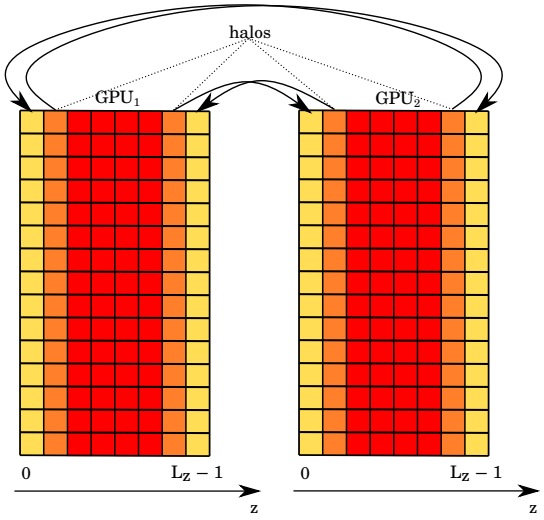

Figure 3. The bulk/halo split scheme in case $N = 2$. The black color (red online) within represents the bulk of the GPUs. These sites can be processed without knowledge about the neighbors. Dark grey (orange online) sites are "halos" – their processing requires data transfer to buffers. Light grey (yellow online) sites are buffers that receive the data from neighbors halos and store it in order to perform $\hat{D}(U)$ action on halos.

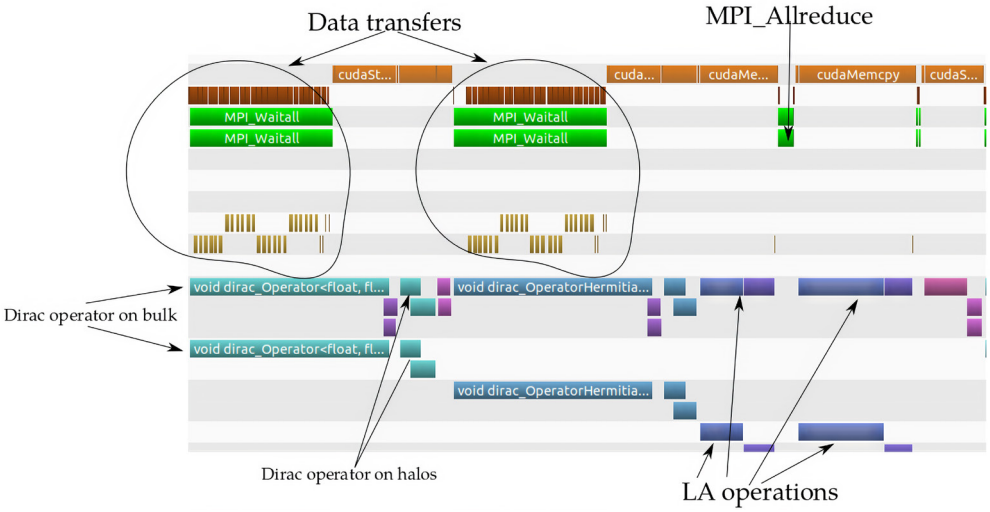

Figure 4. The NVPROF profile of $N = 2$ GPU code executed on the DGX-1 nodes of the "Govorun" supercomputer for the lattice $64^3 \times 16$. The code runs with the CUDA-aware MPI which allows direct P2P copy from GPU to GPU avoiding the CPU.

The Figure 4 shows the NVPROF profile of $N = 2$ GPU code executed on the DGX-1 nodes of the "Govorun" supercomputer for a lattice $64^3 \times 16$. Data transfers lie completely within the Dirac

operator action on bulk. Thus, no additional latency is generated by the memory transfers. The code employs the strategy when every MPI process has only one GPU. This is required in order to run the code on several nodes, Thus, in order to perform scalar products one has to apply the MPI_Allreduce operation. This is the main source of latency for the profile shown in Figure 4.

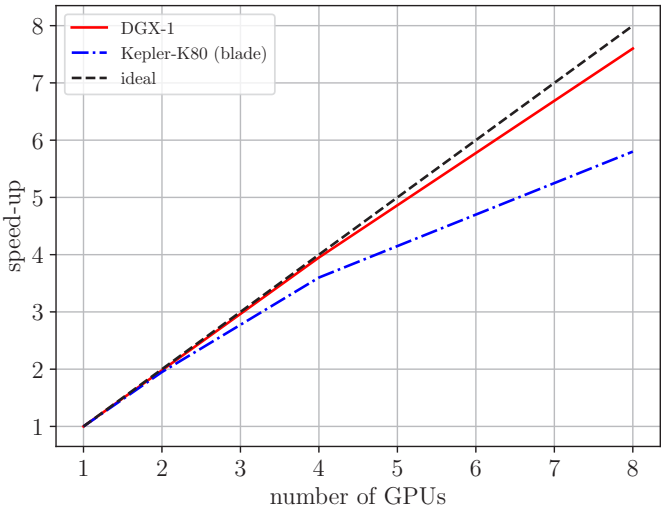

Figure 5. Strong scaling test of N-GPU code speed-up as compared to the single-GPU version. The test was performed on the $64^3 \times 16$ typical contemporary lattice on the K80 "blade" of "HybriLIT" nodes and DGX-1 nodes of "Govorun" supercomputer.

We finally performed the strong and weak scaling tests for the $64^3 \times 16$ typical contemporary lattice on the K80 "blade" of "HybriLIT" nodes and DGX-1 nodes of "Govorun" supercomputer. The results for the strong scaling are shown in Figure 5, while the results for weak scaling do not deviate from perfect, so we do not show them here. On both "blade" and DGX-1 queues the scaling is almost perfect up to 4 GPUs. We then see significant deviation at 8 GPUs for the "blade" queue. The reason it the slow Infiniband internode transfer. This problem is unimportant on two DGX-1 nodes.

5 Discussion and perspectives

The LQCD main algorithms and workflow have been described. The developed multiGPU LQCD code which allows us to significantly speed-up computations and consider much larger lattice volumes was described. The tests performed at the "HybriLIT" platform show that fast CUDA-aware MPI, fast intranode communication on DGX-1 and quite good internode connection on "blade" (K80) allow to perform efficient and scalable computations with small overhead. The created code allows us to start real time simulations connected to the future NICA experiments. The projects include the study of sign-problem-free QCD-like theories such as two-color QCD at finite baryon density [9, 10], $SU(3)$ QCD with isospin density [11] and imaginary baryon chemical potential. We will also study the transport properties of quark-gluon matter such as viscosities [12, 13] and electromagnetic conductivity.

Acknowledgements

This work was supported by RFBR grants 18-32-20172 mol_a_ved and 18-02-40126. This work has been carried out using the "Govorun" supercomputer of the Joint Institute for Nuclear Research.

References

[1] R. Alkofer, Nucl. Phys. Proc. Suppl. **195**, 261 (2009), 0907.5318
[2] T. DeGrand, *Lattice methods for students at a formal TASI*, in *Theoretical Advanced Study Institute in Elementary Particle Physics: The Many Dimensions of Quantum Field Theory (TASI 2019) Boulder, CO, USA, June 3-28, 2019* (2019), 1907.02988
[3] M. D'Elia, Nucl. Phys. **A982**, 99 (2019), 1809.10660
[4] C. Ratti, Rept. Prog. Phys. **81**, 084301 (2018), 1804.07810
[5] J.B. Kogut, Rev. Mod. Phys. **55**, 775 (1983)
[6] T. Takaishi, P. De Forcrand, Physical Review E **73**, 036706 (2006)
[7] C. Bonati, S. Coscetti, M. D'Elia, M. Mesiti, F. Negro, E. Calore, S.F. Schifano, G. Silvi, R. Tripiccione, International Journal of Modern Physics C **28**, 1750063 (2017)
[8] F. Winter, M. Clark, R. Edwards, B. Joó, Proceedings of the International Parallel and Distributed Processing Symposium, IPDPS (2014)
[9] V.V. Braguta, E.M. Ilgenfritz, A.Yu. Kotov, A.V. Molochkov, A.A. Nikolaev, Phys. Rev. **D94**, 114510 (2016), 1605.04090
[10] V.G. Bornyakov, V.V. Braguta, E.M. Ilgenfritz, A.Yu. Kotov, A.V. Molochkov, A.A. Nikolaev, JHEP **03**, 161 (2018), 1711.01869
[11] V.V. Braguta, A.Yu. Kotov, A.A. Nikolaev, JETP Lett. **110**, 1 (2019)
[12] N. Astrakhantsev, V. Braguta, A. Kotov, JHEP **04**, 101 (2017), 1701.02266
[13] N.Yu. Astrakhantsev, V.V. Braguta, A.Yu. Kotov, Phys. Rev. **D98**, 054515 (2018), 1804.02382